CLAIMS

What is claimed is:

1. A method for processing data sequences of arbitrary length in a computing system, the method comprising:

   initializing a load/store buffer by loading a first aligned word of fixed length into the load/store buffer;

   further initializing the load/store buffer by loading a second aligned word into the load/store buffer;

   reading one or more data sequences from the load/store buffer, such that the total length of the sequences in each read does not exceed the fixed length of the first aligned word; and

   loading additional aligned words to the load/store buffer to replace data sequences that are read.

2. The method of claim 1 wherein the data sequence length is a byte.

3. The method of claim 1 wherein the data sequence length is a bit.

4. The method of claim 1 wherein the computing system comprises a processor having an extensible instruction set.

5. The method of claim 1 wherein the computing system comprises a general-purpose processor.

6. The method of claim 1 wherein the first aligned word is stored in a first memory location and the second aligned word is stored in an adjacent second memory location, and the second memory location is accessed by incrementing a memory address pointer after the first aligned word is accessed.

7. The method of claim 1 wherein the first aligned word is stored in a first memory location and the second aligned word is stored in an adjacent second memory location, and the second memory location is accessed by decrementing a memory address pointer after the first aligned word is accessed.

8. A method for processing data sequences of arbitrary length in a computing system, the method comprising:

initializing a load/store buffer by filling the load/store buffer with one or more unaligned data sequences, such that the total length of each data sequence does not exceed the fixed length of an aligned word;

writing one or more unaligned data sequences to the initialized load/store buffer, such that the written unaligned data shifts an aligned word into a memory location; and

flushing the load/store buffer in order to store any of the remaining unaligned data into memory.

9. The method of claim 8 wherein the data sequence length is a byte.

10. The method of claim 8 wherein the data sequence length is a bit.

11. The method of claim 8 wherein the computing system comprises a processor having an extensible instruction set.

12. The method of claim 8 wherein the computing system comprises a general-purpose processor.

13. The method of claim 8 wherein the aligned word is stored in a first memory location and the next aligned word is stored in an adjacent second memory location, and the second memory location is accessed by incrementing a memory address pointer after the first aligned word is accessed.

14. The method of claim 8 wherein the aligned word is stored in a first memory location and the next aligned word is stored in an adjacent second memory location, and the second memory location is accessed by decrementing a memory address pointer after the first aligned word is accessed.

15. A system for processing data sequences of arbitrary length in a computing system, the system comprising:

means for initializing a load/store buffer by loading a first aligned word of fixed length into the load/store buffer;

means for further initializing the load/store buffer by loading a second aligned word into the load/store buffer;

means for reading one or more data sequences from the load/store buffer, such that the total length of the sequences in each read does not exceed the fixed length of the first aligned word; and

means for loading additional aligned words to the load/store buffer to replace data sequences that are read.

16. A system for processing data sequences of arbitrary length in a computing system, the system comprising:

means for initializing a load/store buffer by filling the load/store buffer with one or more unaligned data sequences, such that the total length of each data sequence does not exceed the fixed length of an aligned word;

means for writing one or more unaligned data sequences to the initialized load/store buffer, such that the written unaligned data shifts an aligned word into a memory location; and

means for flushing the load/store buffer in order to store any of the remaining unaligned data into memory.

17. A GET instruction for processing data sequences of arbitrary length in a computing system, the instruction comprising the steps:

initializing a load/store buffer by loading a first aligned word of fixed length into the load/store buffer;

further initializing the load/store buffer by loading a second aligned word into the load/store buffer;

reading one or more data sequences from the load/store buffer, such that the total length of the sequences in each read does not exceed the fixed length of the first aligned word; and

loading additional aligned words to the load/store buffer to replace data sequences that are read.

18. The GET instruction of claim 17 in which the number of data sequences read is an immediate specified number.

19. The GET instruction of claim 17 in which the number of data sequences read is a specified number stored as an index in a register memory.

20. The GET instruction of claim 17 in which a first of the one or more data sequences read is located at a first memory location and the one or more data sequences comprises a specified number of data sequences stored as a first index in a register memory, wherein the subsequent data sequence following the first of the data sequences is located at a second memory location pointed to by a second index.

21. A PUT instruction for processing data sequences of arbitrary length in a computing system, the method comprising:

initializing a load/store buffer by filling the load/store buffer with one or more unaligned data sequences, such that the total length of each data sequence does not exceed the fixed length of an aligned word;

writing one or more unaligned data sequences to the initialized load/store buffer, such that the written unaligned data shifts an aligned word into a memory location; and

flushing the load/store buffer in order to store any of the remaining unaligned data into memory.

22. The PUT instruction of claim 21 in which the number of unaligned data sequences written is an immediate specified number.

23. The PUT instruction of claim 21 in which the number of unaligned data sequences written is a specified number stored as an index in a register memory.